#23

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Paul Freiberger et al.
Assignee:	Interval Research Corporation
Title:	Attention Manager for Occupying the Peripheral Attention of a Person in the Vicinity of a Display Device
Serial No.:	08/620,641 Filed: March 22, 1996
Examiner:	Jeffery A. Brier Group Art Unit: 2775
Attorney Docket No.:	IR-003

Assistant Commissioner for Patents
Washington, D. C. 20231

DECLARATION OF PHILIPPE P. PIERNOT UNDER 37 C.F.R. § 1.131

I, Philippe P. Piernot, hereby declare that:

1. I am an inventor of the invention of the above-referenced patent application.

2. Prior to October 19, 1995, I developed a computer program, an Applescript source code listing of which is attached hereto as Exhibit 1, that, together with the capabilities of conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data as "wallpaper" on a display device of the computer ("content display computer") on which the computer program was executing. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored at a user-designated

location of a non-volatile data storage device of the content display computer. In Exhibit 1, the user-designated location at which content data was stored is indicated at line 5. Line 6 caused execution of a set of instructions (see lines 23-34) that display an image or images generated from the content data. Line 29, together with lines 35-62, caused content data to be retrieved by the content display computer from an appropriate World Wide Web site. In particular, lines 39-41 identified multiple sets of content data to be retrieved (and displayed). Lines 50-54, together with lines 79-~~120,~~ 110 PP caused the sets of content data to be successively retrieved and stored (see, in particular, line 87). Sets of content data were retrieved in alphabetical order of the name of the file containing the content data, in accordance with the manner in which an Applescript computer program orders a list of files within a folder defined on a data storage device (see line 37). Line 30, together with lines 63-78 and lines 134-161, caused identification of the format of a set of content data and display of the set of content data in accordance with the identified format. In the computer program shown in Exhibit 1, sets of content data in either the JPEG format (see lines 140-148) or the GIF format (see lines 150-159) could be used to generate an image display. Lines 31-33 caused the retrieved content data to be used to generate a display of the corresponding image or images: in particular, line 32 caused execution of a computer program called DeskPicture (a commercially available shareware computer program, produced by Peirce Software, that generated a display of an image as

PP

"wallpaper" on a computer display screen) that accessed a set of content data from the appropriate (previously identified; see line 5, discussed above) location on the non-volatile data storage device and produced the corresponding image display. A set of content data was used to generate a display until a new set of content data was to be used to generate a display (the DeskPicture computer program included capabilities for displaying images generated from multiple sets of content data and specifying how long each set of content data was to be used to generate a display of an image), an updated version of the set of content data was to be used to generate a display, or operation of the computer program shown in Exhibit 1 terminated. Lines 10-22 caused the browser software to periodically retrieve (in Exhibit 1, every 5 minutes) and display an updated set of content data corresponding to a set of content data previously retrieved from a Web site. (An updated set of content data could be the same as the corresponding previously retrieved set of content data.)

3. Prior to October 19, 1995, I caused a computer-executable form of the computer program shown in Exhibit 1 to be stored on a first computer ("application management computer"). The application management computer was connected, using conventional hardware and software adapted for such purpose, to a second computer ("content display computer") such that instructions and/or data could be transferred from the application management computer to the content display computer. The presence of the computer-executable version of the computer

program on the application management computer was displayed on a display device of the content display computer. The content display computer was operated in accordance with conventional software that enabled a user of the content display computer to request transfer of the computer program from the application management computer to the content display computer and installation of the computer program on the content display computer. The content display computer was additionally connected, using conventional hardware and software adapted for such purpose, to the Internet computer network, such that the content display computer could be operated in accordance with conventional browser software to enable a user of the content display computer to select an image displayed at a Web site accessible via the Internet computer network so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored at a user-designated location of a non-volatile data storage device of the content display computer.

4. Prior to October 19, 1995, I caused a computer-executable form of a second computer program, similar to the computer program shown in Exhibit 1 (the "first computer program") and having capabilities similar to those described above in paragraph 2 of this Declaration, to be stored on the application management computer discussed above in paragraph 3 of this Declaration. The presence of the computer-executable version of the second computer program on the application management computer was displayed on a display device of the

content display computer. The content display computer discussed above in paragraph 3 of this Declaration was operated in accordance with conventional software that enabled a user of the content display computer to request transfer of the first or second computer program from the application management computer to the content display computer and effect, PB installation of the first or PF second computer program on the content display computer. The second computer program differed from the first computer program in that the types of format of a set of content data that could be displayed were different from the types of format of a set of content data that could be displayed by the first computer program.

5. Prior to October 19, 1995, I developed a computer program, a MacroMedia Director source code listing of which is attached hereto as Exhibit 2, that, together with the capabilities of an Applescript program that I developed (described further below) and conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data as a "screen saver" on a display device of the computer ("content display computer") on which the computer program was executing. The content display computer was operated in accordance with version 7 of the MacIntosh™ operating system. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer and stored

at a user-designated location of a non-volatile data storage device of the content display computer. In Exhibit 2, the user-designated location at which content data was stored is indicated at page 2, line 7. Lines 33-49 on page 6 of Exhibit 2 are a set of instructions that determined whether the screen saver was to be displayed or not. In particular, lines 38-43 prevented the screen saver from being displayed, while lines 45-46 caused the screen saver to be displayed if greater than a specified duration of time (which was user-specified in the computer program shown in Exhibit 2; see line 45 on page 6 of Exhibit 2 and control option 303 in Exhibit 3, discussed below) without interaction with the content display computer (an "idle period") had occurred. Lines 5-32 on page 2 of Exhibit 2 caused the display of one or more images generated from one or more sets of content data. More particularly, lines 5-12 on page 2 of Exhibit 2 determined which set of content data was to be used to generate image(s): each set of content data was used to generate images for a specified amount of time (which was user-specified in the computer program shown in Exhibit 2; see line 5 on page 2 of Exhibit 2 and control option 304 in Exhibit 3, discussed below). Lines 13-30 on page 2 of Exhibit 2 produced an image display from the set of content data identified in lines 5-12. Lines 33-38 on page 2 of Exhibit 2 caused, if appropriate, the screen saver to be turned off again. When the screen saver was turned off, the display shown in Exhibit 3 (discussed below) was produced on the display device of the content display computer using a display screen image definition file as defined using MacroMedia Director constructs adapted for that purpose (see line 37 on page 2 of Exhibit 2). Lines 9-30 on page 6 of Exhibit 2 caused the computer program to periodically retrieve (in the computer program shown in Exhibit 2, within a daily ten minute window beginning at a user-specified time; see lines 10-17 on page 6 of Exhibit 2 and control option 305 in Exhibit 3, discussed below) a set of content data corresponding to Web site image(s) previously selected by a user (see lines 19-23 on page 6 of Exhibit 2). This periodic retrieval of content data occurred only when the screen saver was turned on (see lines 4-8 on page 6 of Exhibit 2, together with the above-mentioned lines 9-30 on page 6 of Exhibit 2). The actual retrieval of content data was accomplished at line 23 using an Applescript computer program called "fetchImages" (which is not shown as part of Exhibit 2) that accessed the user-designated location(s) of the non-volatile data storage device of the content display computer at which content data was stored to identify the World Wide Web site(s) (identification(s), e.g., URL(s), of which were stored together with the corresponding content data) from which the content data was obtained, then caused the browser software to retrieve content data from those site(s). I developed "fetchImages," which embodied the functionality of lines 29, 30, 35-62, 63-78, 79-120 and 134-161 of the computer program shown in Exhibit 1, to enable the Macromedia Director computer program shown in Exhibit 2 to make use of the browser software to transfer set(s) of content data from Web site(s) to the content display computer. (The Macromedia Director computer program shown in Exhibit 2 could not communicate directly with the browser software, but could communicate with an Applescript computer program.)

6. Exhibit 3 depicts a display produced on the display device of the content display computer referred to above in paragraph 5 by the computer program shown in Exhibit 2 (see line 37 on page 2 of Exhibit 2, discussed above) when the screen saver was turned off. The display provided a graphical mechanism for enabling a user of the content display computer to control aspects of the operation of the computer program shown in Exhibit 2. A dialog box (designated by the numeral 301 in Exhibit 3) within the display included four control options that each enabled control of a corresponding aspect of the operation of the computer program shown in Exhibit 2. The first control option (designated by the numeral 302 in Exhibit 3) enabled the user to specify whether the screen saver would be displayed after detection of an idle period. The second control option (designated by the numeral 303 in Exhibit 3) enabled the user to specify the duration of time without interaction with the content display computer which had to pass before the screen saver would be displayed. The third control option (designated by the numeral 304 in Exhibit 3) enabled the user to specify the duration of time for which each set of content data would be used to generate an image display during operation of the screen saver. The fourth control option (designated by the numeral 305 in Exhibit 3) enabled the user to specify the time at which to begin retrieval each day of set(s) of content data corresponding

to Web site image(s) previously selected by a user.

7. Prior to October 19, 1995, I developed a computer program, an Applescript source code listing of which is attached hereto as Exhibit 4, that, together with the capabilities of conventional Internet browser software, acquired content data from a World Wide Web site and displayed an image generated from the content data on a display device of the computer ("content display computer") on which the computer program was executing. The browser software included a capability that allowed a user to select an image displayed at a Web site so as to cause the content data representing the image to be transferred from a data storage device of the Web site to the content display computer. In Exhibit 4, line 4 caused execution of a set of instructions (see lines 21-28) that, in turn, caused the execution of still other sets of instructions to display an image or images generated from the content data. Depending on the type of content data acquired, the image was displayed as "wallpaper" (see line 25 and lines 29-49) or in a display area dedicated to the browser software (see line 26 and lines 50-64). In the former case (i.e., lines 25 and 29-49), lines 44 and 67-89 caused content data to be retrieved by the content display computer for use in generating an image display. After acquisition of the content data, the content data was stored at a user-designated location of a non-volatile data storage device of the content display computer. Lines 46-48 caused the retrieved content data to be used to generate a display of the corresponding image or images: in particular, line 47 caused execution of the computer

program called DeskPicture, as described above in paragraph 2, that produced the image display. In the latter case (i.e., lines 26 and 50-64), the computer program shown in Exhibit 4 did not cause content data to be stored on the non-volatile data storage device of the content display computer, but only used the content data to generate an image display immediately upon acquisition.

8. The acts described above in numbered paragraphs 2 through 7 were carried out in the United States.

9. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: 6/4, 1999 [illegible]

Philippe P. Piernot

```
property justLoaded : false
property folderPath : ""
property triggerMin : 0


on run
	set folderPath to ((path to (the preferences folder)) as string) & "WebPictures:"
	doIt()
	set triggerMin to ((time of (current date)) / minutes) + 5
	set justLoaded to true
end run


on idle
	set mins to (time of (current date)) / minutes
	if mins > triggerMin + 5 then
		if justLoaded then
			set justLoaded to false
		end if
	else
		if not justLoaded and mins ≥ triggerMin then
			doIt()
			set triggerMin to ((time of (current date)) / minutes) + 5
			--set justLoaded to true
		end if
	end if
end idle


on doIt()
	set wasDeskPictureRunning to isProcessRunning("CLY7")
	if wasDeskPictureRunning then
		tell application "DeskPicture"          to quit
	end if

	set fileList to (list folder folderPath)
	fetchAllPicturesIn(folderPath)
	convertToPictAllPicturesIn(folderPath, fileList)

	if wasDeskPictureRunning then
		tell application "DeskPicture"          to run
	end if
end doIt


on fetchAllPicturesIn(folderPath)
	set wasFrontierRunning to isProcessRunning("LAND")

	set fileList to (list folder of folderPath)
	set urlList to {}
	repeat with fileName in fileList
		set urlList to urlList & getFileComment(alias (folderPath & fileName))
```

```
    end repeat

    if not wasFrontierRunning then
        tell application "Frontier"                to quit
    end if

    set wasNetscapeRunning to isProcessRunning("MOSS")
    -- Asks Netscape not to display alert boxes
    tell application "Netscape Navigator™ 3.0"
        set netscapeAlertApp to the alert application
        set alert application to "zzzz"
    end tell

    repeat with i from 1 to (length of fileList)
        set fileName to item i of fileList
        set myURL to item i of urlList
        netscapeGetURL(myURL, (folderPath & fileName & "1"), 5, 300)
    end repeat

    if wasNetscapeRunning then
        -- Resume Netscape alert boxes display handling
        tell application "Netscape Navigator™ 3.0"
            set alert application to netscapeAlertApp
        end tell
    else
        tell application "Netscape Navigator™ 3.0" to quit
    end if
end fetchAllPicturesIn


on convertToPictAllPicturesIn(folderPath, fileList)
    set wasClip2GifRunning to isProcessRunning("c2gf")
    set wasJPegViewRunning to isProcessRunning("JVWR")

    repeat with fileName in fileList
        set fileAlias to (alias (folderPath & fileName))
        convertToPict(folderPath & fileName & "1", fileName & "1")
        «event ScTIExch» (alias (folderPath & fileName & "1")) given «class with»:(fileAlias)
        «event ScTIdele» (alias (folderPath & fileName & "1"))
    end repeat

    if not wasClip2GifRunning then
        tell application "Clip2GiF        to quit
    end if
    if not wasJPegViewRunning then
        tell application "JPegView           to quit
    end if
end convertToPictAllPicturesIn


------------------------NETSCAPE   RELATED   ROUTINES--------------------------------

on netscapeGetURL(myLoc, destFile, nbOfTries, myTimeOut)
    set errCounter to 0
    repeat while errCounter < nbOfTries
```

Exhibit 1

```
        tell application "Netscape Navigator™ 3.0"
            with timeout of myTimeOut seconds
                repeat while the busy of window 1 ≠ 0
                end repeat
                set isLoaded to true
                GetURL myLoc to (file destFile)
                set isLoaded to false
                repeat while not isLoaded
                    try
                        the busy of window 1
                        set isLoaded to true
                    on error
                    end try
                end repeat
            end timeout
            try
                if the file type of (info for (file destFile)) = "TEXT" then
                    set errCounter to errCounter + 1
                    «event ScTldele» destFile
                else
                    return false -- no error
                end if
            on error
                set errCounter to errCounter + 1
            end try
        end tell
    end repeat
    return true -- error
end netscapeGetURL


on «event WWW?PRBG»
    return 1
end «event WWW?PRBG»

on «event WWW?PRMK»
    return 0
end «event WWW?PRMK»

on «event WWW?PREN»
    set finished to true
    return 0
end «event WWW?PREN»


-------------------------FINDER RELATED ROUTINES---------------------------


on isProcessRunning(procString)
    repeat with processName in (list processes)
        if signature of (get process processName) = procString then
            return true
        end if
    end repeat
    return false
end isProcessRunning
```

Page 3

```
on getFileComment(fileAlias)
    tell application "Frontier"
        lfile.getCommentl(fileAlias)
    end tell
end getFileComment


-------------------------PICTURE  CONVERSION  ROUTINE------------------


on convertToPict(filePath, fileName)
    try --We check whether the file exits
        set fileType to the file type of (info for (file filePath))
    on error
        return
    end try
    if fileType = "JPEG" then
        tell application "JPegView"
            try
                open {alias filePath}
                save document 1 in (alias filePath) as picture
                close document 1
            on error
            end try
        end tell
    else
        if fileType = "GIFf" then
            tell application "Clip2Gif
                try
                    open (file filePath) given «class fltp»:picture, «class kfil»:(file (filePath & "2"))
                    «event ScTIdele» (alias filePath)
                    «event ScTIRena» (alias (filePath & "2")) given «class name»:fileName
                on error
                end try
            end tell
        end if
    end if
end convertToPict
```

Exhibit 2

```
on exitFrame
  global gRunning, gLastScreenUpdate

  if desiredScreenSaverState() then
    set gLastScreenUpdate to 0
    initRearWindow()
    savePreferences()
    installMenu              -- removes the menubar
    convertPicturesIfNeeded()
    activate()
    set gRunning to TRUE
    go to frame "SlideShow"
  else
    go to the frame
  end if
end exitFrame
```

Exhibit 2

```
on exitFrame
  global gScreenNumber, gScreenCastNum, gRunning, gLastActivity, gFolderPath,
gLastScreenUpdate

  if desiredScreenSaverState() then
    if (the ticks - gLastScreenUpdate > 60 * value(the text of cast "DisplayTime")) then
      set gScreenNumber to gScreenNumber + 1
      set folderPath to gFolderPath & "Screen Saver Files:"
      set fileName to getNthFileNameInFolder(folderPath, gScreenNumber)
      if fileName = EMPTY then
        set gScreenNumber to 1
        set fileName to getNthFileNameInFolder(folderPath, gScreenNumber)
      end if

      if fileName <> EMPTY then
        if(getFileType(folderPath & fileName) starts "PICT") then
          if the castNum of sprite 2 = 5 then
            set gScreenCastNum to 6
          else
            set gScreenCastNum to 5
          end if
          puppetSprite 2, TRUE
          set the fileName of cast gScreenCastNum to folderPath & fileName
          set pict to the picture of cast gScreenCastNum -- so that the castRect is
updated
          set pict to 0                           -- just in case :-)
          set the castNum of sprite 2 to gScreenCastNum

          set the locH of sprite 2 to (the stageRight - the stageLeft - the width of cast
gScreenCastNum) / 2
          set the locV of sprite 2 to (the stageBottom - the stageTop - the height of
cast gScreenCastNum) / 2
          puppetTransition random(49), 4, 10, FALSE
          set gLastScreenUpdate to the ticks
        end if
      end if
    end if
    go to the frame
  else
    set gRunning to FALSE
    releaseRearWindow()
    installMenu cast "Menubar"
    go to frame "UI"
  end if
end exitFrame
```

Exhibit 2

```
-------------------UTILITY FUNCTIONS--------------------------------------

on filesIn folderPath
  put [] into fileList
  repeat with i = 1 to the maxInteger
    set fileName to getNthFileNameInFolder(folderPath, i)
    if fileName = EMPTY then exit repeat
    append(fileList, fileName)
  end repeat
  return fileList
end filesIn

--------------------------------------------------------------------------

on deleteFile filePath
  set fileIOXObj to FileIO(mNew, "read", filePath)
  return fileIOXObj(mDelete)
end deleteFile

--------------------------------------------------------------------------

on deleteContentOfFolder folderPath
  set fileList to filesIn(folderPath)
  repeat with fileName in fileList
    deleteFile(folderPath & fileName)
  end repeat
end deleteContentOfFolder

--------------------------------------------------------------------------

on newUniqueFileNameIn folderPath
  set counter to -1
  set done to false
  set fileList to filesIn(folderPath)
  repeat while not done
    set counter to counter + 1
    if not getOne(fileList, "" & counter) then
      set done to true
    end if
  end repeat
  return "" & counter
end newUniqueFileNameIn

--------------------------------------------------------------------------

on replaceFilesKeepingComments srcFolderPath, dstFolderPath
  set srcFileList to filesIn(srcFolderPath)
  set dstFileList to filesIn(dstFolderPath)
  repeat with fileName in srcFileList
    if getOne(dstFileList, fileName) then
      set comment to getFileComment(dstFolderPath & fileName)
      deleteFile(dstFolderPath & fileName)
      moveFile(srcFolderPath & fileName, dstFolderPath)
      setFileComment(dstFolderPath & fileName, comment)
    else
      moveFile(srcFolderPath & fileName, dstFolderPath)
    end if
  end repeat
end moveFiles
```

Exhibit 2

```
----------------------------------------------------------------------------

on getFileComment filePath
  set comment to GetComment(filePath)
  set zeroChar to numToChar(0)
  set theLength to the length of comment
  set done to false
  set i to 1
  repeat while not done
    if (i = theLength) or ((char i of comment) = zeroChar) then
      set done to true
    else
      set i to i + 1
    end if
  end repeat
  if i <= 1 then
    return ""
  else
    return char 1 to i - 1 of comment
  end if
end getFileComment

----------------------------------------------------------------------------

on setFileComment filePath, name
  SetComment(filePath, name)
end setFileComment

----------------------------------------------------------------------------

on renameFile filePath, newName
  set oldDelim to the itemDelimiter
  set the itemDelimiter to ":"
  set fileName to the last item of filePath
  set the itemDelimiter to oldDelim
  set folderPathEnd to (the length of filePath) - (the length of fileName)
  set foldPath to (char 1 to folderPathEnd of filePath)

  FSRename(filePath, foldPath & newName)
end renameFile

----------------------------------------------------------------------------

on moveFile filePath, dstFolderPath
  FSCatMove(filePath, dstFolderPath)
end moveFile

----------------------------------------------------------------------------

on getFileType filePath
  set fileIOXObj to FileIO(mNew, "read", filePath)
  set type to fileIOXObj(mGetFinderInfo)
  fileIOXObj(mDispose)
  return type
end getFileType

----------------------------------------------------------------------------
```

Exhibit 2

```
on isProcessRunning procString
  thePrograms "", procString
  return charToNum(char 1 of the result) <> 0
end isProcessRunning

----------------------------------------------------------------------------

on activate
  open the moviePath & the movieName
end activate

----------------------------------------------------------------------------

on getSecondsSinceMidnight
  global gTimeObj

  return gTimeObj(mGetSecsSinceMidnight)
end getSecondsSinceMidnight
```

```
on idle
  global gRunning, gMode, gFolderPath, gFetched

  if gRunning then
    if not(desiredScreenSaverState()) then
      set gRunning to FALSE
      releaseRearWindow()
      installMenu cast "Menubar"
      go to frame "UI"
    else
      set hours to value(the text of cast "hours")
      if the text of cast "am/pm" = "PM" then
        if hours < 12 then
          set hours to hours + 12
        end if
      end if
      set downloadTime to (3600 * hours) + (60 * value(the text of cast "minutes"))
      if gFetched = 0 and gMode = "Done" and getSecondsSinceMidnight() > downloadTime and
¬
getSecondsSinceMidnight() < (downloadTime + 600) then
        set gFetched to the ticks
        deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
        deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
        deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
        set gMode to "FetchAndConvert"
        open the moviePath & "Helper Apps:fetchImages"
      else
        if gFetched <> 0 and the ticks - gFetched > 36000 then -- we should be done
downloading
          set gFetched to 0
        end if
      end if
    end if
  end if
  pass
end idle


on desiredScreenSaverState
  global gLastActivity, gLastMouseH, gLastMouseV, gLastKeyCode, gKeyDetectorXObj

  set mH to the mouseH
  set mV to the mouseV
  set kc to the keyCode
  if not (the hilite of cast "on/off") or the mouseDown or mH <> gLastMouseH or mV <>
gLastMouseV ¬ or gLastKeyCode <> kc or gKeyDetectorXObj(mCheckKey) <> 0 then
    set gLastMouseH to mH
    set gLastMouseV to mV
    set gLastKeyCode to kc
    set gLastActivity to the ticks
    return FALSE
  else
    if the ticks - gLastActivity > 3600 * value(the text of cast "SleepDelay") then
      return TRUE
    end if
  end if
end desiredScreenSaverState
```

Exhibit 2

```
on startMovie
  global gMode, gTimeObj, gKeyDetectorXObj, gMiscXObj, gLastScreenUpdate,
gScreenDisplayTime, ¬
gScreenNumber, gRunning, gFolderPath, gFetched

  set gScreenNumber to 0
  set gFetched to 0
  set gRunning to FALSE
  set gMode to "Done"
  set the hilite of cast "on/off" to TRUE
  set gLastScreenUpdate to 0
  set gScreenDisplayTime to 600

  set gTimeObj to TimeSinceMidnight( mNew )
  set gKeyDetectorXObj to KeyDetector(mNew)
  set gMiscXObj to misc_x(mNew)

  set gFolderPath to gMiscXObj(mPrefsFolder) & "NetScreen:"
  installMenu cast "Menubar"
  loadPreferences()

  --put callBackFactory(mNew) into callbackObject
  --setCallBack RunOSAScript, callbackObject
  --RunOSAScript("open")
end startMovie


on stopMovie
  global gTimeObj, gKeyDetectorXObj, gMiscXObj

  savePreferences()
  if objectP(gTimeObj) then
    gTimeObj(mDispose)
  end if
  if objectP(gKeyDetectorXObj) then
    gKeyDetectorXObj(mDispose)
  end if
  if objectP(gMiscXObj) then
    gMiscXObj(mDispose)
  end if
  releaseRearWindow()

  --RunOSAScript("close")
  --callBackFactory(mDispose)
end stopMovie


on convertPicturesIfNeeded
  global gMode, gFolderPath

  if gMode = "Done" then
    set files to filesToConvert()
    if files <> EMPTY then
      deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
      deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
      deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
      repeat with fileName in files
        moveFile(gFolderPath & "Screen Saver Files:" & fileName, ¬
gFolderPath & "Temporary Files:downloaded:")
```

```
      end repeat
      set gMode to "Convert"
      open the moviePath & "Helper Apps:fetchImages"
    end if
  end if
end convertPicturesIfNeeded

on filesToConvert
  global gFolderPath

  set folderPath to gFolderPath & "Screen Saver Files:"
  set fileList to filesIn(folderPath)
  set files to []
  repeat with fileName in fileList
    set type to getFileType(folderPath & fileName)
    if not (type starts "PICT") then
      append files, fileName
    end if
  end repeat
  return files
end filesToConvert

on quitNetScreen
  stopMovie()
  quit
end quitNetScreen

----------------------------------------------------------------------------

on getStatus
  global gFolderPath, gMode

  if voidP(gMode) then
    set gMode to "Done"
  end if
  set folderPath to gFolderPath & "Screen Saver Files:"
  set status to gMode & " " & ¬
  isProcessRunning("MOSS") & " " & ¬
  isProcessRunning("c2gf")
  if gMode = "FetchAndConvert" then
    set fileList to filesIn(folderPath)
    repeat with fileName in fileList
      set status to status & RETURN & fileName & RETURN & ¬
getFileComment(folderPath & fileName)
    end repeat
  end if
  return status
end getStatus

----------------------------------------------------------------------------

on ScriptDone
  global gFolderPath, gMode

  if gMode = "FetchAndConvert" then
    replaceFilesKeepingComments(gFolderPath & "Temporary Files:converted:", ¬
gFolderPath & "Screen Saver Files:")
  else
    if gMode = "Convert" then
```

```
      set files to filesIn(gFolderPath & "Temporary Files:downloaded:")
      repeat with fileName in files
        set comment to getFileComment(gFolderPath & "Temporary Files:downloaded:" &
fileName)
        setFileComment(gFolderPath & "Temporary Files:converted:" & fileName, comment)
      end repeat
      replaceFilesKeepingComments(gFolderPath & "Temporary Files:converted:", ¬
gFolderPath & "Screen Saver Files:")
    end if
  end if
  deleteContentOfFolder(gFolderPath & "Temporary Files:downloaded:")
  deleteContentOfFolder(gFolderPath & "Temporary Files:temp:")
  deleteContentOfFolder(gFolderPath & "Temporary Files:converted:")
  set gMode to "Done"
  activate()
end ScriptDone

on loadPreferences
  global gFolderPath

  set prefPath to gFolderPath & "NetScreen.prefs"
  set fileXObj to FileIO(mNew, "read", prefPath)
  set l to fileXObj(mReadLine)
  set the hilite of cast "on/off" to value(word 2 of l)
  set l to fileXObj(mReadLine)
  set the text of cast "SleepDelay" to word 2 of l
  set l to fileXObj(mReadLine)
  set the text of cast "DisplayTime" to word 2 of l
  set l to fileXObj(mReadLine)
  set the text of cast "hours" to word 2 of l
  set the text of cast "minutes" to word 3 of l
  set the text of cast "am/pm" to word 4 of l
  fileXObj(mDispose)
end loadPreferences


on savePreferences
  global gFolderPath

  set prefPath to gFolderPath & "NetScreen.prefs"
  set fileXObj to FileIO(mNew, "write", prefPath)
  fileXObj(mWriteString, "on/off " & the hilite of cast "on/off" & RETURN)
  fileXObj(mWriteString, "SleepDelay " & the text of cast "SleepDelay" & RETURN)
  fileXObj(mWriteString, "DisplayTime " & the text of cast "DisplayTime" & RETURN)
  fileXObj(mWriteString, "DownloadTime " & the text of cast "hours" & " " & ¬
the text of cast "minutes" & " " & the text of cast "am/pm" & RETURN)
  fileXObj(mDispose)
end savePreferences

  -- Factory: MISC_X ID:10001
  -- Misc_X, Misc Utils XObject, v1.1.3
  --I    mNew
  --S    mBootName
  --S    mSystemFolder
  --S    mPrefsFolder
  --IS   mFileExists, fP
  --ISS  mCopyFile, sP, dP
  --IS   mFolderExists, fP
  --IS   mInsureFolder, fP
```

Exhibit 2

```
195  --XS    mDeleteFolder, fP.
196  --SS    mFolderList, fP
197  --SSSSS mAsk, q, dR, bOk, bCan
198  --SSSSS mAnswer, q, bL, bM, bR
199  --IS    mSpaceOnVol, vN
200  --X     mFlushActions
```

Exhibit 2

```
global gRwObj

on initRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if

  if createRwObject() >= 0 then
    gRwObj(mPatToWindow, -5)                    -- Paint in back
  end if
end initRearWindow

on releaseRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if
end releaseRearWindow

on createRwObject
  if not objectP(gRwObj) then
    -- "M" indicates multiple monitors, "S" is for single monitor configuration.
    -- ONLY use "S" if there is not enough room for multiple monitors.
    -- So first...let's try it with multiple-monitor configuration:
    set gRwObj = RearWindow(mNew, "M")
    set error to value(gRwObj)
    if error < 0 then
      gRwObj(mDispose)
      return error
    end if
    if the freeBlock < gRwObj(mGetMemoryNeeded) then
      -- delete the object and create it again with a single-monitor config...
      if objectP(gRwObj) then
        gRwObj(mDispose)
        set gRwObj = RearWindow(mNew, "S")
      end if
      set error to value(gRwObj)
      if error < 0 then
        gRwObj(mDispose)
        return error
      end if
    end if
  end if
  return value(gRwObj)
end createRwObject
```

Exhibit 2

```
global gRwObj

on initRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if

  if createRwObject() >= 0 then
    gRwObj(mPatToWindow, -5)                    -- Paint in back
  end if
end initRearWindow

on releaseRearWindow
  if objectP(gRwObj) then
    gRwObj(mDispose)
  end if
end releaseRearWindow

on createRwObject
  if not objectP(gRwObj) then
    -- "M" indicates multiple monitors, "S" is for single monitor configuration.
    -- ONLY use "S" if there is not enough room for multiple monitors.
    -- So first...let's try it with multiple-monitor configuration:
    set gRwObj = RearWindow(mNew, "M")
    set error to value(gRwObj)
    if error < 0 then
      gRwObj(mDispose)
      return error
    end if
    if the freeBlock < gRwObj(mGetMemoryNeeded) then
      -- delete the object and create it again with a single-monitor config...
      if objectP(gRwObj) then
        gRwObj(mDispose)
        set gRwObj = RearWindow(mNew, "S")
      end if
      set error to value(gRwObj)
      if error < 0 then
        gRwObj(mDispose)
        return error
      end if
    end if
  end if
  return value(gRwObj)
end createRwObject
```

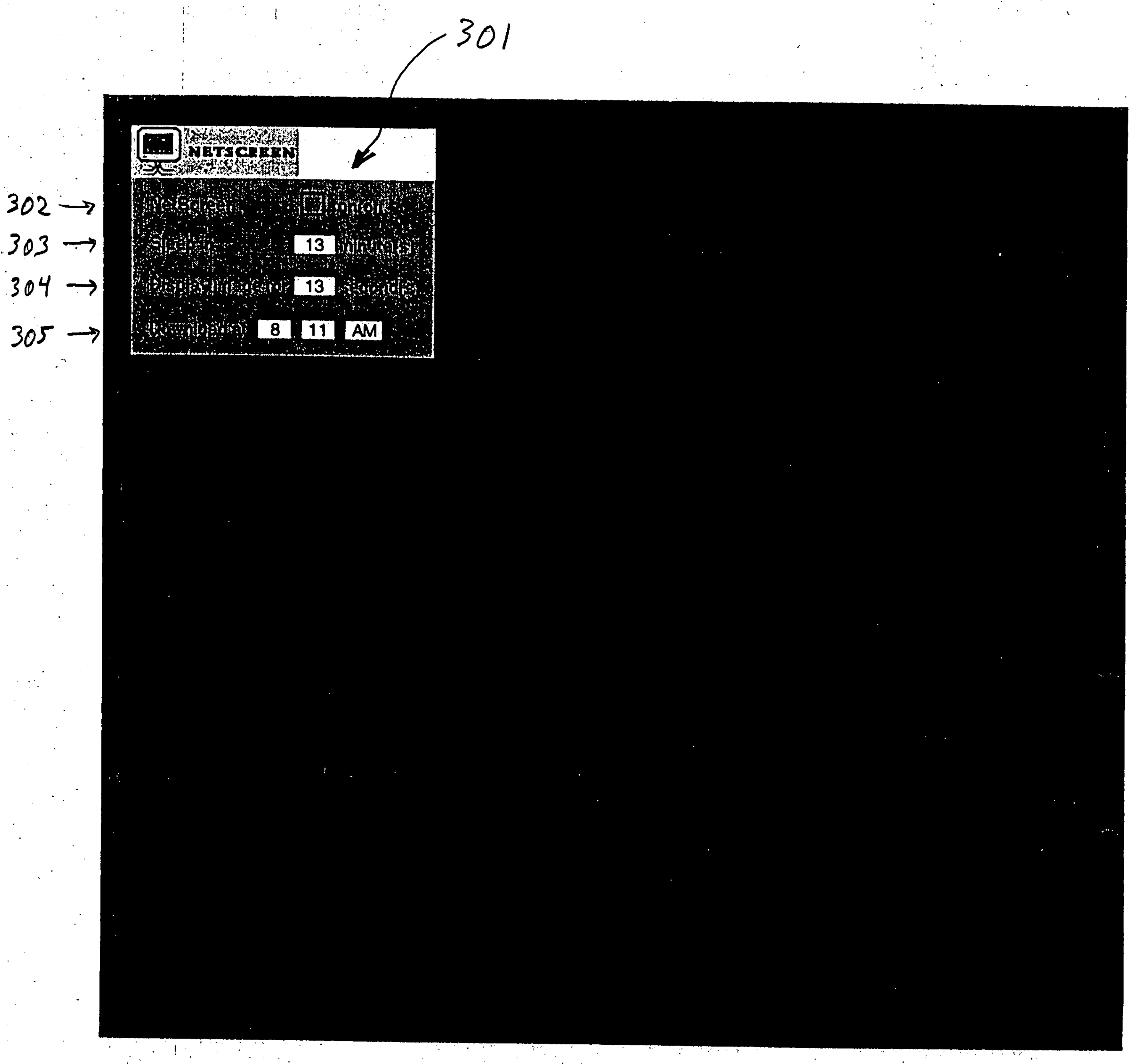
Exhibit 2

```
--factory callBackFactory
--method mNew
--   me(mPut, 1, "SendCardMessage")
--   me(mPut, 2, "EvalExpr")
--   me(mPut, 3, "StringLength")
--   me(mPut, 4, "StringMatch")
--   me(mPut, 5, "SendHCMessage")
--   me(mPut, 6, "ZeroBytes")
--   me(mPut, 7, "PasToZero")
--   me(mPut, 8, "ZeroToPas")
--   me(mPut, 9, "StrToLong")
--   me(mPut, 10, "StrToNum")
--   me(mPut, 11, "StrToBool")
--   me(mPut, 12, "StrToExt")
--   me(mPut, 13, "LongToStr")
--   me(mPut, 14, "NumToStr")
--   me(mPut, 15, "NumToHex")
--   me(mPut, 16, "BoolToStr")
--   me(mPut, 17, "ExtToStr")
--   me(mPut, 18, "GetGlobal")
--   me(mPut, 19, "SetGlobal")
--   me(mPut, 20, "GetFieldByName")
--   me(mPut, 21, "GetFieldByNum")
--   me(mPut, 22, "GetFieldByID")
--   me(mPut, 23, "SetFieldByName")
--   me(mPut, 24, "SetFieldByNum")
--   me(mPut, 25, "SetFieldById")
--   me(mPut, 26, "StringEqual")
--   me(mPut, 27, "ReturnToPas")
--   me(mPut, 28, "ScanToReturn")
--   me(mPut, 31, "FormatScript")
--   me(mPut, 32, "ZeroTermHandle")
--   me(mPut, 33, "PrintTEHandle")
--   me(mPut, 34, "SendHCEvent")
--   me(mPut, 35, "HCWordBreakProc")
--   me(mPut, 36, "BeginXSound")
--   me(mPut, 37, "EndXSound")
--   me(mPut, 38, "RunHandler")
--   me(mPut, 39, "ScanToZero")
--   me(mPut, 40, "GetXResInfo")
--   me(mPut, 41, "GetFilePath")
--   me(mPut, 42, "FrontDocWindow")
--   me(mPut, 43, "PointToStr")
--   me(mPut, 44, "RectToStr")
--   me(mPut, 45, "StrToPoint")
--   me(mPut, 46, "StrToPoint")
--   me(mPut, 47, "GetFieldTE")
--   me(mPut, 48, "SetFieldTE")
--   me(mPut, 49, "GetObjectName")
--   me(mPut, 50, "GetObjectScript")
--   me(mPut, 51, "SetObjectScript")
--   me(mPut, 52, "StackNameToNum")
--   me(mPut, 53, "Notify")
--   me(mPut, 54, "SowHCAlert")
--   me(mPut, 100, "NewXWindow/GetNewXWindow")
--   me(mPut, 101, "CloseXWindow")
--   me(mPut, 102, "SetXWIdleTime")
--   me(mPut, 103, "XWHasInterruptCode")
--   me(mPut, 104, "RegisterXWMenu")
--  me(mPut, 105, "BeginXWEdit/EndXWedit")
--  me(mPut, 106, "SaveXWScript")
--  me(mPut, 107, "GetCheckPoints")
--  me(mPut, 108, "SetCheckPoint")
--  me(mPut, 109, "XWAllowReEntrancy")
--  me(mPut, 110, "SendWindowMessage")
--  me(mPut, 111, "HideHCPalettes")
--  me(mPut, 112, "ShowHCPalettes")
--  me(mPut, 113, "XWAlwaysMoveHigh")
--  me(mPut, 200, "GoScript")
--  me(mPut, 201, "StepScript")
--  me(mPut, 202, "AbortScript")
--  me(mPut, 203, "CountHandlerInfo")
--  me(mPut, 204, "GetHandlerInfo")
--  me(mPut, 205, "GetVarInfo")
--  me(mPut, 206, "SetVarValue")
--  me(mPut, 207, "GetStackCrawl")
--  me(mPut, 208, "TraceScript")
--
--method mEvalExpr x
--  --  put "mEvalExpr" && x
--  --  if x = "cd fld " & QUOTE & "urlField" & QUOTE then
--  --    return "tell application " & QUOTE & "Netscape" & QUOTE & " to make new window"
--  --  else
--  --    if x = "the name of cd fld " & QUOTE & "urlField" & QUOTE then
--  --      put "beep"
--  --      return "urlField"
--  --    else
--  --      if x = "the id of cd fld " & QUOTE & "urlField" & QUOTE then
--  --        put "beep beep"
--  --        --return 100
--  --      end if
--  --    end if
--  --  end if
--  if word 1 of x = "----" then
--    return "tell me to activate"
--  end if
--end mEvalExpr
--
--method mEvalExpr x
--  put "mEvalExpr" && x
--  if the length of x >= 10 then
--    set s to char 1 to 10 of x
--    if (s <> "the id of ") and (s <> "the name o") then
--      return x
--    end if
--  end if
--end mEvalExpr
--
--method mSendHCMessage x
--put "mSendHCMessage" && x
--
--method mSendCardMessage x
--put "mSendCardMessage" && x
--
--method mGetFieldByName card, name
--put "mGetFieldByName" && card && name
--
--method mGetFieldByNum card, Num
```

Exhibit 2

```
--put "mGetFieldByNum" && card && num
--
--method mGetFieldByID card, id
--put "mGetFieldByID" && card && id
--
--method mSetFieldByName card, name, value
--put "mSetFieldByName" && card && name && value
--
--method mSetFieldByNum card, num, value
--put "mSetFieldByNum" && card && num && value
--
--method mSetFieldByID card, id, value
--put "mSetFieldByID" &7 card && id && value
--
--method mGetFieldTE
--put "mGetFieldTE" --&& arg1 && arg2 && arg3
--
--method mUnknown which
--put me(mGet, value(which)) into callBackName
--put "mUnknown:" && which && "(" & ¬
--  callBackName & ")"
--
--
```

Exhibit 3



NetScreen ☐ On/Off

Sleep in 13 minutes

Display image for 13 seconds

Download at 8 11 AM

Exhibit 4

```
property justLoaded : false
property folderPath : ""


on run
    doIt()
    set justLoaded to true
end run


on idle
    set mins to (time of (current date)) / minutes
    set triggerMin to 0
    if mins > triggerMin + 30 then
        if justLoaded then
            set justLoaded to false
        end if
    else
        if not justLoaded and mins ≥ triggerMin then
            doIt()
            set justLoaded to true
        end if
    end if
end idle


on doIt()
    if folderPath = "" then
        set folderPath to ((path to (the preferences folder)) as string) & "WebTrio Documents:"
    end if

    doDesktopDisplay()
    doNetscapeDisplay()
    doScreenSaverDisplay()
end doIt


on doDesktopDisplay()
    set wasDeskPictureRunning to isProcessRunning("CLY7")
    if wasDeskPictureRunning then
        tell application "DeskPicture" to quit
    end if

    set wasFrontierRunning to isProcessRunning("LAND")

    set fileList to (list folder of (folderPath & "For the Desktop:"))
    set urlList to {}
    repeat with fileName in fileList
        set urlList to getFileComment(alias (folderPath & "For the Desktop:" & fileName)) & urlList
    end repeat

    if not wasFrontierRunning then
```

Exhibit 4

```
        tell application "Frontier"   to quit
    end if

    set fileList to (list folder (folderPath & "For the Desktop:"))
    fetchAllPictures(urlList, folderPath & "For the Desktop:", fileList)
    convertToPictAllPicturesIn(folderPath & "For the Desktop:", fileList)

    if wasDeskPictureRunning then
        tell application "DeskPicture" to run
    end if
end doDesktopDisplay


on doNetsacapeDisplay()
    set wasFrontierRunning to isProcessRunning("LAND")

    set fileList to (list folder of (folderPath & "For Netscape"))
    set urlList to {}
    repeat with fileName in fileList
        set urlList to urlList & getFileComment(alias (folderPath & "For Netscape:" & fileName))
    end repeat

    if not wasFrontierRunning then
        tell application "Frontier"   to quit
    end if

    tell application "Netscape Navigator™ 3.0"
        make new document
    end tell

    fetchAllPictures(folderPath & "For Netscape:", "", false)
end doNetsacapeDisplay


on doScreenSaverDisplay()
end doScreenSaverDisplay


on fetchAllPictures(urlList, folderPath, fileList)
    set wasNetscapeRunning to isProcessRunning("MOSS")
    tell application "Netscape Navigator™ 3.0"
        set netscapeAlertApp to the alert application
        set alert application to "zzzz" -- Asks Netscape not to display alert boxes
    end tell

    repeat with i from 1 to (length of urlList)
        set myURL to item i of urlList
        if folderPath ≠ "" then
            set fileName to item i of fileList
            netscapeGetURL(myURL, (folderPath & fileName & "1"), 5, 300)
        else
            netscapeGetURL(myURL, "", 5, 300)
        end if
    end repeat

    if wasNetscapeRunning then
```

Page 2

```
            tell application "Netscape Navigator™ 3.0"
                set alert application to netscapeAlertApp -- Resume Netscape alert boxes display handling
            end tell
        else
            tell application "Netscape Navigator™ 3.0" to quit
        end if
    end fetchAllPictures


    on convertToPictAllPicturesIn(folderPath, fileList)
        set wasClip2GifRunning to isProcessRunning("c2gf")
        set wasJPegViewRunning to isProcessRunning("JVWR")

        repeat with fileName in fileList
            set fileAlias to (alias (folderPath & fileName))
            convertToPict(folderPath & fileName & "1", fileName & "1")
            «event ScTIExch» (alias (folderPath & fileName & "1")) given «class with»:(fileAlias)
            «event ScTIdele» (alias (folderPath & fileName & "1"))
        end repeat

        if not wasClip2GifRunning then
            tell application "clip2gif" to quit
        end if
        if not wasJPegViewRunning then
            tell application "JPegView"   to quit
        end if
    end convertToPictAllPicturesIn


    ------------------------NETSCAPE   RELATED   ROUTINES---------------------------------


    on netscapeGetURL(myLoc, destFile, nbOfTries, myTimeOut)
        set errCounter to 0
        repeat while errCounter < nbOfTries
            tell application "Netscape Navigator™ 3.0"
                with timeout of myTimeOut seconds
                    repeat while the busy of window 1 ≠ 0
                    end repeat
                    set isLoaded to true
                    GetURL myLoc to (file destFile)
                    set isLoaded to false
                    repeat while not isLoaded
                        try
                            the busy of window 1
                            set isLoaded to true
                        on error
                        end try
                    end repeat
                end timeout
                try
                    if the file type of (info for (file destFile)) = "TEXT" then
                        set errCounter to errCounter + 1
                        «event ScTIdele» destFile
                    else
                        return false -- no error
                end if
            on error
                set errCounter to errCounter + 1
            end try
        end tell
    end repeat
    return true -- error
end netscapeGetURL


on «event WWW?PRBG»
    return 1
end «event WWW?PRBG»

on «event WWW?PRMK»
    return 0
end «event WWW?PRMK»

on «event WWW?PREN»
    set finished to true
    return 0
end «event WWW?PREN»


--------------------------FINDER RELATED ROUTINES---------------------------


on isProcessRunning(procString)
    repeat with processName in (list processes)
        if signature of (get process processName) = procString then
            return true
        end if
    end repeat
    return false
end isProcessRunning


on getFileComment(fileAlias)
    tell application "Frontier"
        Ifile.getCommentI(fileAlias)
    end tell
end getFileComment


-------------------------PICTURE CONVERSION ROUTINE------------------


on convertToPict(filePath, fileName)
    try --We check whether the file exits
        set fileType to the file type of (info for (file filePath))
    on error
        return
    end try
    if fileType = "JPEG" then
        tell application "JPegView"
            try
```

```
                open {alias filePath}
                save document 1 in (alias filePath) as picture
                close document 1
            on error
            end try
        end tell
    else
        if fileType = "GIFf" then
            tell application "clip2gif"
                try
                    open (file filePath) given «class fltp»:picture, «class kfil»:(file (filePath & "2"))
                    «event ScTldele» (alias filePath)
                    «event ScTlRena» (alias (filePath & "2")) given «class name»:fileName
                on error
                end try
            end tell
        end if
    end if
end convertToPict
```